AFRL-RI-RS-TR-2008-203
Final Technical Report
July 2008

UNIFIED PLATFORM-INDEPENDENT AIRBORNE NETWORKING ARCHITECTURE FOR VIDEO COMPRESSION

University of California, Berkeley

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the Air Force Research Laboratory Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2008-203 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/ /s/

DAVID HENCH
Work Unit Manager

WARREN H. DEBANY, JR
Technical Advisor
Information Grid Division

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JUL 2008	Final	MAR 07 – DEC 07

4. TITLE AND SUBTITLE

UNIFIED PLATFORM-INDEPENDENT AIRBORNE NETWORKING ARCHITECTURE FOR VIDEO COMPRESSION

5a. CONTRACT NUMBER

5b. GRANT NUMBER
FA8750-07-2-0070

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

Kannan Ramchandran

5d. PROJECT NUMBER
BATC

5e. TASK NUMBER
DI

5f. WORK UNIT NUMBER
SV

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California, Berkeley
2150 Shattuck Ave. Room 313
Berkeley, CA 94704-5940

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RIGC
525 Brooks Rd
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RIGC

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2008-203

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# WPAFB 08-3891

13. SUPPLEMENTARY NOTES

14. ABSTRACT

It is highly desirable to have a video coding framework that can flexibly distributed coding complexity between the video encoder and decoder. However, today's video compression techniques impose a rigid computational complexity distribution between the video encoder and decoder. Specifically, video decoders are very simple but encoders are computationally complex as they need to carry out motion estimation and mode decision. Distributed video coding has shown great potentials in enabling flexible complexity distribution between the video encoder and decoder. Previous work on distributed video coding has studied two extreme operating points of complexity distribution: 1) Complex encoder with light-weight decoder to achieve maximum compression efficiency and 2) Light encoder and complex decoder to achieve robustness against transmission packet drops (at the cost of reduced compression efficiency). In this project, our main goal is to explore the middle ground between these two extreme points. In particular, we study the effect of doing coarse motion search at the encoder.

15. SUBJECT TERMS
video compression, video encoding, video decoding

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			David Hench
U	U	U	UU	26	19b. TELEPHONE NUMBER (*Include area code*) 315-330-4540

Table of Contents

Summary

To target a heterogeneous range of video encoding and decoding platforms, it is highly desirable to have a video coding framework that can flexibly distributed coding complexity between the video encoder and decoder. However, today's state-of-the-art video compression techniques impose a rigid computational complexity distribution between the video encoder and decoder. Specifically, video decoders are very simple but encoders are computationally complex as they need to carry out motion estimation and mode decision. There have been a number of recent works on video coding using distributed source coding (DSC) principles, often abbreviated as distributed video coding. They have shown great potentials in enabling flexible complexity distribution between the video encoder and decoder.

In our previous work on distributed video coding, we have studied two extreme operating points of complexity distribution:

1. Complex encoder with light-weight decoder to achieve maximum compression efficiency given a transmission channel condition, and
2. Light encoder and complex decoder to achieve robustness against transmission packet drops (at the cost of reduced compression efficiency).

In this project, our main goal is to explore the middle ground between these two extreme points. In particular, we study the effect of doing coarse motion search at the encoder. By doing coarse motion search at the encoder, we would like to achieve two goals:

1. Reduce the decoding complexity, and
2. Improve rate-distortion performance.

As a high-level summary, we investigated the effect of having motion search of various levels at the encoder, implemented hierarchical coarse motion search at the encoder, improved the encoder classifier to improve rate-distortion performance, and changed the decoding process to make use of the motion search result at the encoder.

We have the following main findings:

1. Not all sequences can benefit from encoder motion search under the current framework. Sequences that can benefit from encoder motion search are the ones with a certain level of motion intensity.
2. For sequences with a certain level of motion intensity, complexity tradeoff between the encoder and the decoder can be achieved. When the encoder carries out proper hierarchical motion search, the decoding motion search complexity can be reduced by 50-80%, such as in football, Stefan and garden sequences.
3. If the sequence also has smooth motion, up to 1 dB gain in RD performance can be achieved. This is because the decoder is now able to find a much better quality predictor (or side

information) using the coarse motion vector provided by the encoder. This is not true for sequences with irregular motion, such as football sequence. This is because the gain in video quality is offset by the much higher rate used to encode motion vector compared to sequences with smoother motion.

4. When there are transmission packet drops, in general, the higher the packet drop rate is, the more decoder motion searches are needed and the lower the decoded PSNR will be. For sequences with very irregular motion, such as football, the motion vectors become useless when the motion-compensated predictor is not correctly reconstructed because the motion vector provides very little clue on where to find a best predictor two frames ago. This results in drastically increased decoding complexity for such sequences as packet drop rates goes up.

Introduction

Current video compression technologies based on motion-compensated predictive coding (MCPC), that are part of video coding standards like MPEG and H.264, have a complexity distribution that is somewhat rigid, namely a *complex* encoder and a *light* decoder. These systems derive their compression efficiency by using computationally intensive motion-estimation at the encoder, the dominant complexity component in the system.

This architecture made sense when the primary drivers of video compression technology were TV broadcast and heavy-server-to-light-client video download. However, with the current push towards a much wider range of encoding platforms, ranging from camera-phones to surveillance cameras to heterogeneous UAV platforms, video coding systems are being increasingly hampered by the inflexibility of the MPEG architecture, which demands computationally complex encoders. Specifically, the MPEG architecture is not well suited for the setup where the encoders is battery-constrained, such as a mobile phone and low-power surveillance wireless camera, whereas the decoder is a much more capable Pentium-powered computing unit or a high-end PC.

Simultaneously, with the proliferation of wireless networks, there is high demand that video be transmitted reliably over channels characterized by severe fades, packet drops, and temporary channel outages. In particular, the Airborne Network will not only have much higher bit error rates as compared to a wired link but also highly variable error rate due to distance, fading, and the influence of EMI. These channels severely impair the performance of MCPC-based systems like MPEG, which rely on successful transmission of all the motion compensated prediction differences between the current frame and the previous frame for effective compression. When the previous frame is decoded erroneously (due to channel losses), the encoder and decoder get out of synch, creating mismatch or *drift* between the two ends (see Figure 1). As MPEG-style architectures rely on a large dependency prediction chain for their performance, this drift accumulates for each succeeding frame, resulting in potentially catastrophic video quality degradation.

We aim to design and develop a fundamentally new flexible and unified video coding architecture to:

- Accommodate a wide range of computational platforms at encoder and decoder; and
- Design up-front for robustness and in-built immunity to transmission channel noise.

We propose to move away from the conventional predictive coding framework of MPEG and H.264, and instead adopt a new paradigm built on principles of distributed source coding (DSC) from multi-user information theory. Distributed source coding based video coding, often abbreviated as distributed video coding, has been an active area of research. It has shown great potentials in a wide range of applications, such as low-complexity encoding, robust real-time video transmission, scalable video and multiple-camera video coding. In this work, we focus on two specific benefits of the distributed video coding framework: flexible distribution of complexity, and robust real-time video transmission.

In our previous work on distributed video coding, we have studied two extreme operating points of complexity distribution:

1. Complex encoder with full motion search and light-weight decoder (Wang, Prabhakaran, & Ramchandran, 2006): achieve maximum compression efficiency given a transmission channel condition, and
2. Light encoder with no motion search and complex decoder (Puri & Ramchandran, PRISM: A New Robust Video Coding Architecture Based on Distributed Compression Principles, 2002), (Puri, Majumdar, & Ramchandran, 2007): achieve robustness against transmission packet drops (at the cost of reduced compression efficiency).

In this project, we aim to explore the middle ground between these two extreme points. In particular, we study the effect of doing coarse motion search at the encoder. By doing coarse motion search at the encoder, we would like to achieve two goals:

1. Reduce the decoding complexity, thus achieving complexity re-distribution between the encoder and the decoder;
2. Improve rate-distortion performance.

Methods, Assumptions, and Procedures

We first give a brief review of the main results of distributed source coding. We will then review the details of the distributed video coding framework that we built upon. Finally we will detail the changes we have made to the framework to achieve flexible complexity distribution while maintaining robustness during transmission.

Distributed source coding background

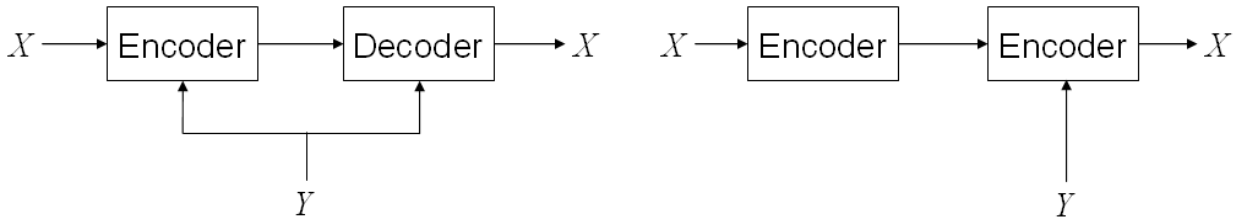


Figure 1: X and Y are correlated sources. X is the source that the encoder wants to transmit. Y is a correlated source (side-information), available only at the decoder. (a) Both encoder and decoder access and use Y in order to compress X. (b) Only the decoder accesses Y. Using distributed source coding principles, it is possible to compress X as efficiently in this case as in case (a).

Consider the problems depicted in Figure 1. The goal is to compress and send source X at the lowest rate possible. Y is another source that is correlation to X. In Figure 3(a), the side-information Y is available only to the decoder, while in Figure 3(b) it is available to both encoder and decoder. From information theory (Cover & Thomas) we know that the rate region for the problem of Figure 3(a), when the side-information is available to both encoder and decoder, is $R \geq H(X|Y)$, the conditional entropy of X given Y. The surprising result of Slepian and Wolf (Slepian & Wolf, 1973) is that the rate region for the problem of Figure 3(b), when the side-information is available only to the decoder, is also $R \geq H(X|Y)$. The overall achievable rate region is:

$$R_X \geq H(X|Y)$$

$$R_Y \geq H(Y|X)$$

$$R_X + R_Y \geq H(X,Y)$$

At the high level, in a video coding setup, one can imagine X to be the current video frame being encoded and Y to be the previous decoded frame. The case where Y is only available at the decoder is precisely the case where the previous frame gets corrupted during transmission. Since the encoder does not know exactly which packets are corrupted, it has no way of knowing what the decoded frame will look like. The Slepian-Wolf theorem tells us that as long as we can characterize the statistical correlation between the current frame X and the previous decoded frame Y available at the decoder only, we will still be able to correctly reconstruct X at the decoder with an encoding rate that is the *same as when Y is available at the*

encoder. These results were extended to the lossy case by Wyner-Ziv (Wyner & Ziv, 1976) a few years later (for the case when Y is known perfectly at the decoder). Again, X and Y are two correlated random variables. The problem here is to decode X to its quantized reconstruction X' given a constraint on the distortion measure $E[d(X; X')]$ when the side information Y is available only at the decoder. In the special case where the sources are memoryless Gaussian and when MSE is used as the distortion measure, the Wyner-Ziv theorem shows that we can do just as well when Y is only available at the decoder. Further, in (Pradhan, Chou, & Ramchandran, 2003) it was proved that for $X = Y + N$, only the innovations N needs to be Gaussian for this result to hold.

Review of PRISM

We now review the encoding and decoding processes of our previous work that we built upon.

Figure 1 shows encoding block diagram of the existing PRISM encoder. For a detailed description of the encoding process, please refer to (Puri, Majumdar, & Ramchandran, 2007). The key to its low encoding complexity is the *zero-motion* classifier, which determines the mode of an 8x8 block using its zero-motion mean square error (MSE). This mode will then dictate how many DCT coefficients are to be syndrome-encoded and what syndrome code strength is needed for each of these coefficients.

To decode each block, the decoder needs to find the right predictor. This search starts from the co-located block in the previous frame and spirals outwards. In theory, the decoder can carry out syndrome decoding and filters out the wrong predictor using joint typicality check. However, in practice, this will not work due to the short block length of the syndrome code. Instead, we use the CRC codes (a hash of the original block) to detect whether syndrome decoded result is correct. To summarize, for each candidate predictor, the decoder will first syndrome decode and then carry out CRC check. If the check fails, the decoder will choose a different predictor. If the check passes, this predictor is a valid one and the decoder stops searching (Figure 2). The syndrome decoded result can be used to reconstruct the original block. The main complexity at the decoder comes from motion search followed by syndrome decoding. The need for CRC also increases the encoding bit rate significantly.

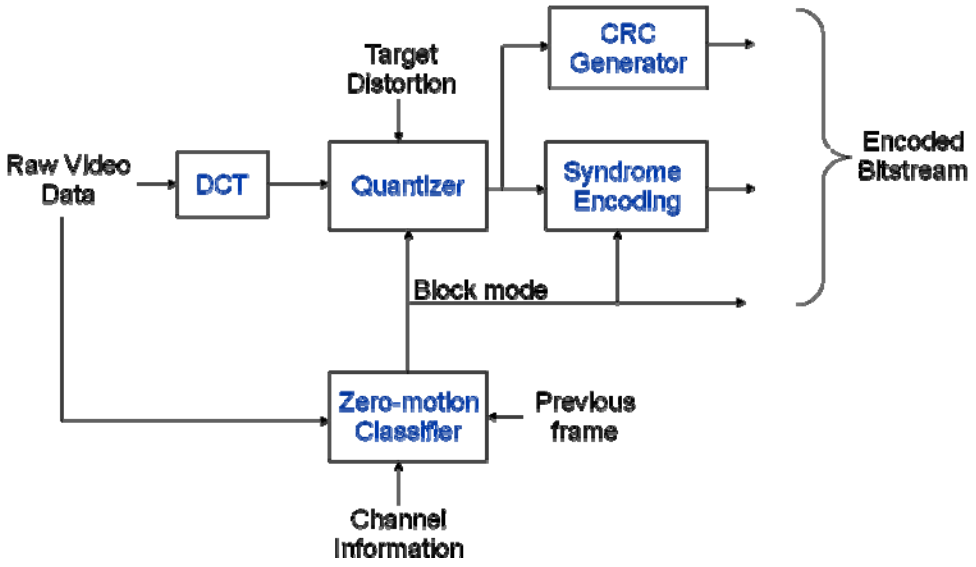


Figure 2: Block diagram of previous PRISM encoder

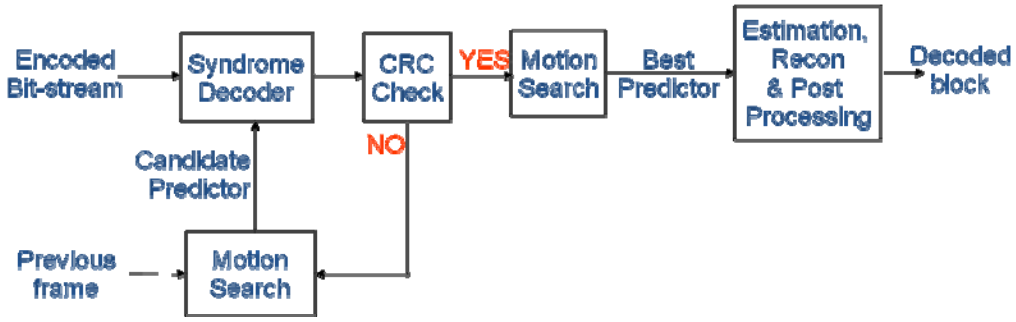


Figure 3: Block diagram of previous PRISM decoder

Proposed method

We will now details the changes we have made to the encoder and decoder to enable complexity shift between the encoder and decoder and to improve the rate-distortion performance of the system.

Encoder

Changes at a glance:

- Investigated the effectiveness of motion search of various granularities on reducing decoding complexity
- Implemented hierarchical motion search
- Implemented differential coding (within a slice or packet) of motion vectors
- Reduced CRC rate from 16 bit per block to 8 bit per block
- Retrained classifier

Details of the changes:

1. **Investigation of the effectiveness of motion search of various granularities on reducing decoding complexity**

The first investigation we carried out was to study the effect of having access to accurate motion vectors at the decoder. To do this, we carried out full motion search at the encoder and transmitted the motion vectors to the decoder. To decode each block, instead of doing a spiral search starting from the co-located block in the previous frame, the decoder will now start the spiral search from the motion compensated predictor. We assume a clean transmission channel and all the data packets are received at the decoder.

We tested a few standard sequences including Foreman, Football, Flower Garden and Stefan, all in CIF format (352x288). We encoded the first 15 frames (1 GOP, I-P-P-P structure) of each sequence using I-frame quantization step size 5 and P-frame quantization step size 8.

Table 1: Study of effect of full encoder motion search on reducing decoding complexity

Motion intensity	Sequence	Encoder search method	Number of encoder motion search	Number of decoder motion search	PSNR (dB)
Low	Foreman	Zero motion	0	4198	36.04
		Full motion	20004442	4139	36.36
Medium	Flower Garden	Zero motion	0	15784	34.93
		Full motion	20004694	10929	35.55
High	Stefan	Zero motion	0	101890	35.47
		Full motion	20005160	20402	35.85
	Football	Zero motion	0	35438	35.77
		Full motion	20006491	10028	35.84

From the results summarized in Table 1, we see that motion vectors can indeed help reduce decoder motion search complexity. The amount of reduction highly depends on the motion intensity of the video sequence. Intuitively, in low-motion videos, the co-located block is oftentimes the best predictor, thus any more motion search at the encoder will not help. For medium to high-motion videos, on the other hand, the co-located block is rarely a good predictor for the current block. Hence a large number of decoder motion searches are needed to find a qualified predictor.

2. **Implemented hierarchical motion search**

While we would like to shift the motion search operations from the decoder to the encoder, we are still interested in keeping the encoder complexity as low as possible without sacrificing decoder complexity or decoding quality. Thus we adopt the hierarchical fast motion search algorithm to reduce encoder complexity. The hierarchical motion search we adopted consists of the following steps.

Step 1: Motion search of 4-pixel accuracy within ± 16 pixels of the location of the current block.
Step 2: Motion search of 2-pixel accuracy around the predictor found by Step 1 within ± 8 pixels.
Step 3: Motion search of 1-pixel accuracy around the predictor found by Step 2 within ± 4 pixels.

Step 4: Find the MSE between the current block and the motion compensated predictor found by Step 3. If MSE < threshold (20000 in current implementation), use this motion vector. Otherwise, discard motion vector and carry out 1-pixel accuracy full motion search ± 16 pixels of the location of the current block.

Note that this is a simplified version of the more sophisticated 3-step motion search, which requires filtering the original picture and subsample twice to get QCIF and QQCIF version of the original sequence (assuming CIF sequence). It terms of the total number of motion searches needed at the encoder, this simplified version is not as good as the more sophisticated one. But it does not require sampling or sub-sampling.

Table 2 compares the number of encoder motion search between the hierarchical search and the full search. We see that hierarchical search takes only a fraction of the complexity that full motion search requires. But at the decoder, the number of decoder motion searches needed and the decoded PSNR are barely changed, indicating the quality of the predictors found by the hierarchical search is almost as good as the full search.

An interesting finding is that Step 4 is crucial in this hierarchical motion search. To flexibly shift complexity between the encoder and the decoder, we would like to reduce the encoder motion search complexity and study the effect at the decoder. Ideally, we would like to see increased decoder complexity but not higher complexity than when there is no motion search at the encoder.

Given the hierarchical motion search procedures, the seemingly most logical approach is to eliminate steps backwards. However, an interesting and surprising observation is that Step 5 (the refinement step) is actually crucial to performance. The reason is that fast motion search may occasionally completely miss a good predictor and mistreat an outlier as a reasonable predictor. For sequences with intensive and unpredictable motion, there are more outliers. When this happens, the best predictors are typically quite far away from the outlier and up to more than 1000 decoder motion searches are needed to correct a single block. The refinement step works perfectly to identify these outliers and eliminate them. The following table represents this phenomenon. Here we will focus on the medium to high motion sequences as motion searches are more important to them. We can see that for Stefan and Football sequences, if the refinement step is skipped, the number of decoder motion searches needed is actually even higher than when no motion vectors are provided at the decoder at all. On the other hand, Flower Garden sequence has very smooth and predictable motion due to smooth camera panning therefore there aren't many outliers. As a result, even though Flower Garden sequence has a certain level of motion intensity, hierarchical motion search works quite well even without the refinement step.

Table 2: Study of effectiveness of motion estimation of various granularities in reducing decoding complexity

Motion intensity	Sequence	Encoder search method	Number of encoder motion search	Number of decoder motion search	PSNR (dB)

High	Stefan	Zero motion	0	101890	35.47
		Hierarchical without refinement step	3108807	127991	35.76
		Hierarchical	4076492	20742	35.84
		Full search	20005160	20402	35.85
	Football	Zero motion	0	35438	35.77
		Hierarchical without refinement step	3119210	40903	35.77
		Hierarchical	5119268	10076	35.83
		Full search	20006491	10028	35.84
Medium	Flower Garden	Zero motion	0	15784	34.93
		Hierarchical without refinement step	3090263	11588	35.55
		Hierarchical	4016450	10929	35.55
		Full search	20004694	10929	35.55

3. Implemented differential coding (within a slice or packet) of motion vectors

In the current implementation, the predictor motion vector is that of the block to the left of the current block. If the block to the left is INTRA or SKIP-coded, the motion vector of that block is assumed to be zero. If a block is the left-most block of the row, the original value is stored. The Huffman table used for motion vectors in H.263 is used for the entropy coding of the differential motion vectors.

4. Reduced CRC rate from 16 bit per block to 8 bit per block

Due to the presence of the motion vector, the decoder now has much better knowledge on where to find an appropriate predictor (side information). The decoder can typically find a good predictor within a couple searches while without motion vectors, the decoder sometimes needs thousands of searches before finding a good predictor. This means that with motion vectors, the number of error patterns the decoder could encounter due to decoding off of a bad predictor is much limited. As a result, a much weaker CRC is needed. We find that 8-bit CRC suffice through experiments.

Note: a 16-bit CRC can identify 2^{16} errors while an 8-bit CRC can only identify 2^{8} errors.

5. Retrained classifier

The classifier contains the following information

- Threshold for MSE (in dB) between current block and reference block to determine which class the current block belongs to.
- For each class
    - The number of coefficients to syndrome encode.

- The mean and variance of each coefficient that is to be syndrome-encoded (Note: The variance will determine the strength of the channel code needed for the coefficient. The mean is used only at the decoder to get estimation gain.)

In our previous work, the MSE between the current block and the co-located block in the previous frame is computed. We call this zero-motion MSE. The classifier was also trained offline to threshold the zero-motion MSE. In this work, the MSE between the current block and the motion-compensated block in the previous frame is computed, which we call motion-compensated MSE. Clearly, a zero-motion MSE and a motion-compensated MSE of the same value do not have the same meaning, i.e. the best predictor of a block whose motion-compensated MSE is the same as another block's zero-motion MSE is likely to be worse than the best predictor of that block. This calls for retraining the classifier with motion compensated MSE. The training was done in the following way:

- For each block, take DCT of both the current block and the motion-compensated predictor block. For each DCT coefficient, find the difference between the current block and the predictor block, which we term DFD (displaced frame difference).
- For each block, record its motion-compensated MSE . Then for each DCT coefficient of this block, record the value of each coefficient and the value of DFD.
- Plot the histogram of motion-compensated MSE (in dB).
- Divide the range of MSE values into 16 regions/classes (one SKIP, one INTRA, 14 PRISM modes), i.e. 15 thresholds. The lowest threshold (in dB value) is the INTRA threshold while the highest threshold is the SKIP threshold. These two values are set based on heuristics. The other 13 thresholds are set such that the number of blocks in each of the 14 classes is roughly equalized.
- Group blocks by their class. For each class of blocks, find the mean and variance of each DCT coefficient and DFD.

Decoder

Changes at a glance:

1. Implemented motion vector differential decoding
2. Implemented motion search centered around *appropriately* motion-compensated predictor instead of co-located previous block

Details of the changes:

1. Implemented motion vector differential decoding

This part is straight-forward. The decoding is done corresponding to the encoding.

2. Implemented motion search centered around *appropriately* motion-compensated predictor instead of co-located previous block

When there are no transmission packet drops, this is relatively straight-forward. For the decoder motion search, the search spirals and will start at the motion-compensated predictor in the previous frame.

However, when there are transmission packet drops, this needs to be modified depending on the error concealment strategy used. This is because when there are packet drops in the previous frame and the motion-compensated predictor is part of the missing packet, how the missing blocks are filled up will determine where best to start the motion search. In the current implementation, if a packet (a slice of 16x16 macroblocks) is lost, the co-located slice from the previous frame is pasted for error concealment. Using this concealment strategy, it is no longer a good idea to start the decoder motion search at the motion-compensated predictor if that predictor was part of a lost packet because the block at that location was pasted from 2 frames ago and is oftentimes not a good predictor unless the motion for the current block is zero. To better understand this, let us consider the example in Figure 3. Clearly in this case, if the decoder search still starts at the position pointed to by the motion vector, i.e. two blocks to the left, it will not find the best predictor right away. In fact, using spiral search, it will take a large number of searches before reaching the best predictor. To solve this problem, the decoder needs to track the decoding status of each block, i.e. whether it was lost, correctly reconstructed, or failed to decode. When the decoder starts searching, it will first look at the status of the motion-compensated predictor. If it's correctly reconstructed[1], then the decoder motion search will start from there. Otherwise, some modifications are needed. In the current implementation, the decoder will

[1] A block is considered correctly reconstructed if 1) it is an INTRA block and is received, or 2) it is a PRISM block and it was received and successfully decoded, or 3) it is a SKIP block and the block it is pasting from was correctly reconstructed.

interpolate the motion vector and start motion search from the new motion-compensated predictor in the previous reconstructed frame[2].

One can see that if the concealment strategy is different, this method will need to be modified. For instance, if when filling in the missing blocks, motion vectors of neighboring blocks that are received are used to paste in motion-compensated blocks, we would not need to interpolate the motion vectors. However, this method is only effective when the lost packets do not contain neighboring slices. For a bursty channel where a few packets often get lost at the same time, this method may not work well.

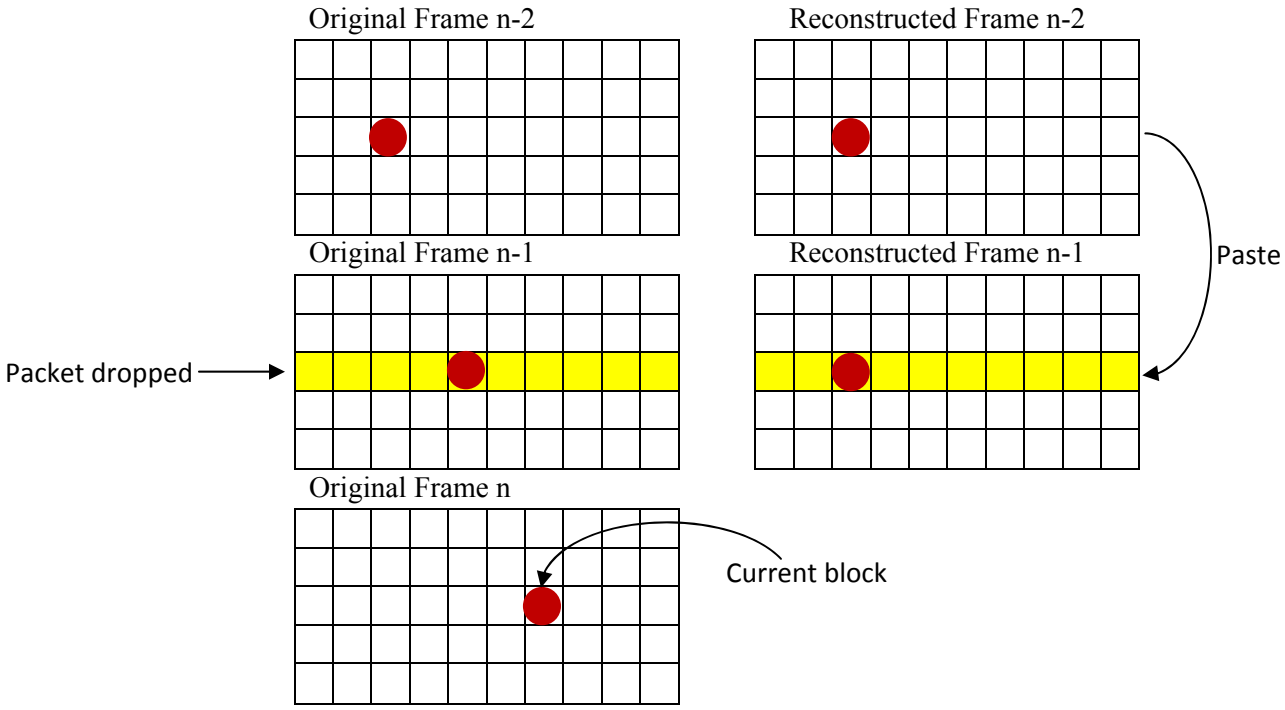


Figure 4: Illustration of proper decoder motion search

[2] Ideally, the decoder should search in Frame n-2 using the interpolated motion vectors. But this requires a an additional frame buffer. The current strategy works well if horizontal motion dominates, which is typically the case.

Results and Discussion

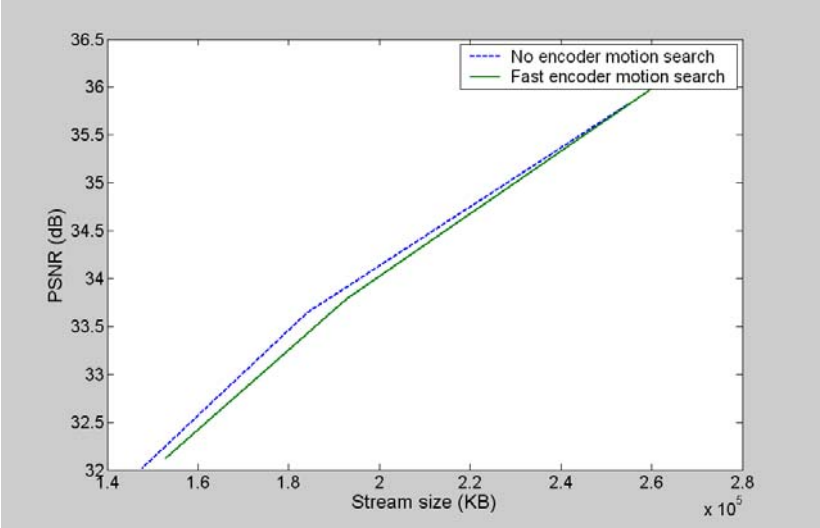
Based on our study, we have the following main findings:

- Not all sequences can benefit from encoder motion search under the current framework. Sequences that can benefit from encoder motion search are the ones with a certain level of motion intensity.
- For sequences with a certain level of motion intensity, complexity tradeoff between the encoder and the decoder can be achieved. When the encoder carries out hierarchical motion search, the decoding motion search complexity can be reduced by 50-80%, such as in football, Stefan and garden sequences.
- If the sequence also has smooth motion, up to 1 dB gain in RD performance can be achieved due to being able to find a much better quality predictor (or side information) at the decoder using motion vector. This is not true for sequences with irregular motion, such as football sequence, because the rate used to encode motion vector is much higher than that of the sequences with smoother motion.
- When there are transmission packet drops, in general, the higher the packet drop rate, the more decoder motion searches are needed and the lower the PSNR. For sequences with very irregular motion, such as football, the motion vectors become useless when the motion-compensated predictor is not correctly reconstructed because the motion vector provides very little clue on where to find a best predictor two frames ago. This results in drastically increased decoding complexity for such sequences as packet drop rates goes up.

1. Results using the first 30 frames of standard sequences with no transmission packet drops
    a. Foreman sequence: 30 frames, 2 GOPs

QP	Size (KB)	PSNR	Number of encoder motion search	Number of decoder motion search
8	254738	35.82	0	5512
	262178	36.05	10116948	6855
12	184143	33.65	0	3258
	192796	33.79	10116948	3986
16	147722	32.02	0	1982
	152726	32.12	10116948	2398

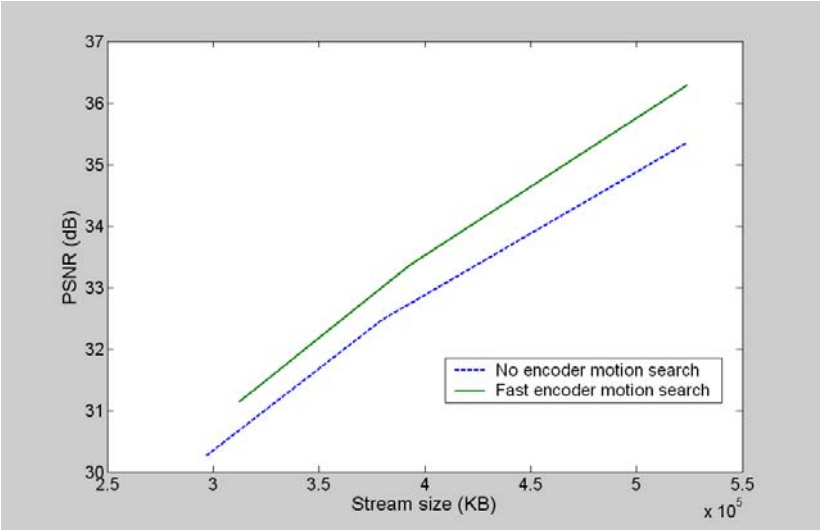
RD curve



b. Stefan sequence: 30 frames, 2 GOPs

QP	Size (KB)	PSNR	Number of encoder motion search	Number of decoder motion search
8	523302	35.34	0	126519
	524045	36.29	15728564	17985
12	379618	32.47	0	76451
	392499	33.36	15728564	14984
16	296295	30.25	0	44391
	312183	31.13	15728564	12213

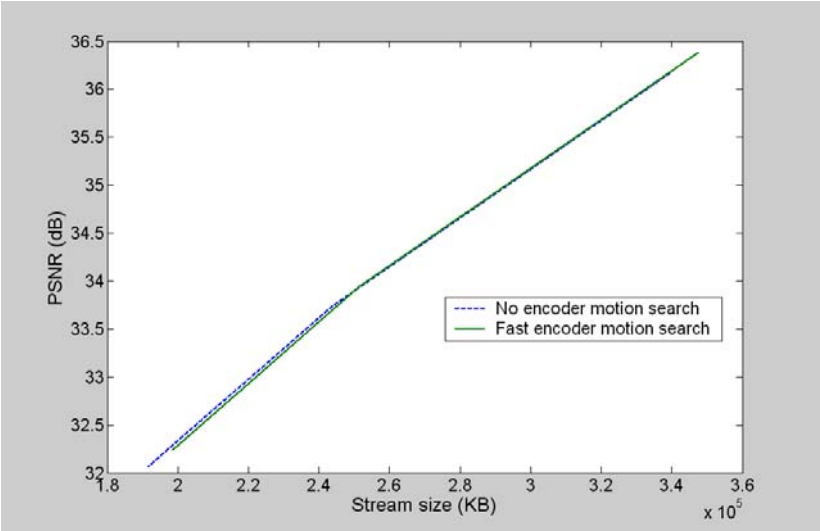
RD curve



c. Football sequence: 30 frames, 2 GOPs

QP	Size (KB)	PSNR	Number of encoder motion search	Number of decoder motion search
8	339435	36.17	0	27844
	347724	36.39	17334404	5107
12	243739	33.74	0	24365
	251231	33.94	17334404	3419
16	191427	32.06	0	10882
	198385	32.24	17334404	2589

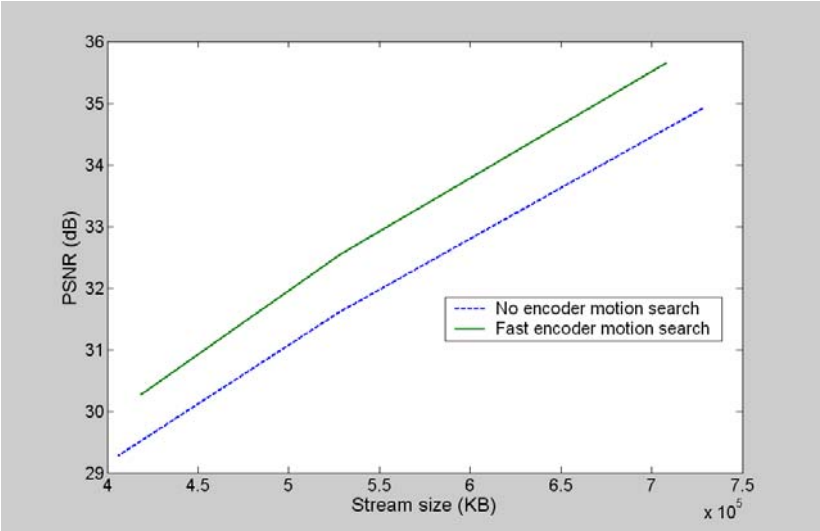
RD curve



d. Garden sequence: 30 frames, 2 GOPs

QP	Size (KB)	PSNR	Number of encoder motion search	Number of decoder motion search
8	728193	34.92	0	34296
	708378	35.65	18666260	17890
12	527708	31.61	0	24378
	528170	32.54	18666260	15710
16	405896	29.28	0	17641
	418198	30.26	18666260	13756

RD curve



2. Results using first 30 frames of standard sequences with transmission packet drops

a. Stefan sequence

	Packet drop rates (%)	0	2.58	4.17	5.75	7.74	9.72
Zero encoder motion search	Number of decoder searches	126519	128601	132144	166212	328967	366458
	PSNR	35.34	33.42	32.84	31.64	30.26	28.83
Coarse encoder motion search	Number of decoder searches	17985	62641	76634	101331	130414	175897
	PSNR	36.29	33.11	32.31	30.55	28.96	27.69

b. Flower garden sequence

	Packet drop rates (%)	0	2.58	4.17	5.75	7.74	9.72
Zero encoder motion search	Number of decoder searches	34296	34842	35753	36690	38852	40514
	PSNR	34.92	33.61	32.5	31.54	30.43	29.8
Coarse encoder motion search	Number of decoder searches	17890	18201	18242	18369	37522	37761
	PSNR	35.65	34.19	33.01	31.99	30.22	29.59

c. Tempete sequence

	Packet drop rates (%)	0	2.58	4.17	5.75	7.74	9.72
Zero encoder motion search	Number of decoder searches	20945	49266	104603	123195	268678	330238
	PSNR	34.38	33.62	33.09	32.44	31.98	31.22
Coarse encoder motion search	Number of decoder searches	16105	17374	17091	16602	20706	22935
	PSNR	34.44	33.69	33.27	32.63	32.14	31.38

d. Football sequence

	Packet drop rates (%)	0	2.58	4.17	5.75	7.74	9.72
Zero encoder motion search	Number of decoder searches	27844	34913	36773	45272	47478	49171
	PSNR	36.17	34.23	32.65	31.36	29.25	28.13
Coarse encoder motion search	Number of decoder searches	5107	16091	34724	47664	72493	86793
	PSNR	36.38	34.29	32.49	31.14	28.92	27.81

Conclusions

In this project, we investigated the possibility of flexibly distribute computational complexity between the video encoder and decoder. We propose to carry out coarse motion search at the encoder in the distributed video coding frame. The goal is to reduce decoding complexity and to improve rate-distortion performance without sacrificing robustness to transmission packet loss.

We have the following main findings:

1. Not all sequences can benefit from encoder motion search under the current framework. Sequences that can benefit from encoder motion search are the ones with a certain level of motion intensity.
2. For sequences with a certain level of motion intensity, complexity tradeoff between the encoder and the decoder can be achieved. When the encoder carries out proper hierarchical motion search, the decoding motion search complexity can be reduced by 50-80%, such as in football, Stefan and garden sequences.
3. If the sequence also has smooth motion, up to 1 dB gain in RD performance can be achieved. This is because the decoder is now able to find a much better quality predictor (or side information) using the coarse motion vector provided by the encoder. This is not true for sequences with irregular motion, such as football sequence. This is because the gain in video quality is offset by the much higher rate used to encode motion vector compared to sequences with smoother motion.
4. When there are transmission packet drops, in general, the higher the packet drop rate is, the more decoder motion searches are needed and the lower the decoded PSNR will be. Decoder motion search needs to be modified significantly to make proper use of the coarse motion vectors. For sequences with very irregular motion, such as football, the motion vectors become useless when the motion-compensated predictor is not correctly reconstructed because the motion vector provides very little clue on where to find a best predictor two frames ago. This results in drastically increased decoding complexity for such sequences as packet drop rates goes up.

References

Cover, T. M., & Thomas, J. A. *Elements of Information Theory.* New York: John Wiley and Sons.

Pradhan, S. S., Chou, J., & Ramchandran, K. (2003). Duality Between Source Coding and Channel Coding and its Extension to the Side Information Case. *International Transactions on Information Theory* , 1181-2003.

Puri, R., & Ramchandran, K. (2002). PRISM: A New Robust Video Coding Architecture Based on Distributed Compression Principles. *Allerton Conference on Communication, Control, and Computing.* Urbana-Champaign, IL.

Puri, R., Majumdar, A., & Ramchandran, K. (2007). PRISM: A Video Coding Paradigm With Motion Estimation at the Decoder. *IEEE Transactions on Image Processing* , 2436-2448.

Slepian, D., & Wolf, J. K. (1973). Noiseless Coding of Correlated Information Sources. *IEEE Transactions on Information Theory* , 471-480.

Wang, J., Prabhakaran, V., & Ramchandran, K. (2006). Syndrome-based robust video transmission over networks with bursty losses. *International Conference on Image Processing.* Atlanta, GA.

Wyner, A. D., & Ziv, J. (1976). The Rate-Distortion Function for Source Coding with Side Information at the Decoder. *IEEE Transactions on Information Theory* , 1-10.

List(s) of Symbols, Abbreviations, and Acronyms

- MCPC: motion-compensated predictive coding
- DSC: distributed source coding
- RD: rate-distortion
- PRISM: name of the distributed video coding codec we built upon
- PSNR: peak signal-to-noise ratio
- $H(X)$: entropy of X
- $H(X|Y)$: conditional entropy of X given Y
- $H(X,Y)$: joint entropy of X and Y
- $E[\cdot]$: expected value